GITAM
DEEMED TO BE UNIVERSITY

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GITAM SCHOOL OF TECHNOLOGY

## GITAM (Deemed to be University)

# A CASE STUDY ON
# DIET TRACKING SYSTEM(CLI Based)


| Registration No(s) | Name(s) | Signature |
|---|---|---|
| 2024064631 | B Lohith Srikar | |
| 2024010949 | Nikhil Behra | |
| 2024145238 | Ch Tarun | |


# DECLARATION

**We hereby declare that the project report entitled** Diet Tracking System(CLI Based) in Python is an original work done in the Department of Computer Science and Engineering, GITAM School of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B Tech in Computer Science and Engineering/Computer Science and Engineering (CS) The work has not been submitted to any other college or University for the award of any degree or diploma

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
# GITAM SCHOOL OF TECHNOLOGY

## GITAM (Deemed to be University)

# CERTIFICATE

This is to certify that the project report entitled "Diet Tracking System(CLI Based) in Python" is a bonafide record of work carried out by B.Lohith Srikar(2024064631), Nikhil Behra(2024010949),Ch.Tarun(2024145238) students submitted in partial fulfillment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering / Computer Science and Engineering, Cyber Security.

DATE:

Project Head

Head of Department

# “DIET TRACKING SYSTEM(CLI Based)”

A Project Report submitted in partial fulfillment of the requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY**

IN COMPUTER SCIENCE AND ENGINEERING (or) COMPUTER SCIENCE AND ENGINEERING (Specialisation in Cyber Security)

**SUBMITTED BY**

**B.Lohith Srikar(2024064631)**
**Nikhil Behra(2024010949)**
**Ch.Tarun(2024145238)**

Under the esteemed guidance of **Sujatha Varadi**


DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GITAM
SCHOOL OF TECHNOLOGY GITAM (Deemed to be University)

VISAKHAPATNAM

# DIET TRACKING SYSTEM

*Case Study by*

B Lohith Srikar
Nikhil Behra
Ch Tarun

# Table of Content

# INTRODUCTION

# 1.1 ABSTRACT

Calorie tracking is a system for monitoring daily food intake and managing dietary goals. It helps individuals keep track of their calories, ensuring they maintain a balanced diet based on their needs.
In a calorie tracker, users log their meals, and the system calculates their total calorie intake. Nutritional information is fetched from a database, helping users make informed dietary choices. The system also allows users to set weight loss or gain targets and provides recommendations accordingly.

This project is something similar to a calorie tracking system, where users can create a profile, enter their personal details, and receive customized calorie intake suggestions. This project is completely user-friendly, allowing users to log meals, view nutritional details, and track their progress easily.

To use the service, a user first needs to register by providing their name, age, gender, height, weight, and activity level. The system then calculates their daily calorie needs and helps them maintain their diet accordingly.

# 1.2 PROBLEM STATEMENT

Many individuals struggle to maintain a balanced diet due to the lack of an efficient and accurate way to track their daily calorie intake. Manual calorie tracking is often **time-consuming, error-prone, and inconvenient.**

Additionally, existing solutions may:

- Require manual data entry, making them tedious.
- Lack real-time access to reliable nutritional information.
- Fail to provide personalized insights based on user goals.

# 1.3 UNDERSTANDING THE PROBLEM

Calorie tracking involves recording daily food intake and analyzing nutritional values to monitor energy consumption. This process requires accurate data retrieval, meal categorization, and automated calorie calculation. Traditional methods rely on manual logging, which is time-consuming and prone to errors.

By integrating an API-based system, users can efficiently fetch nutritional information, log meals under different categories, and track their progress over time. The output should provide accurate calorie estimates and personalized insights to help users make informed dietary choices.

# 1.4 CONCEPTS USED IN THE CODE

The Diet Tracking System applies various programming concepts to automate calorie tracking efficiently.

**File Handling & Data Persistence:**

- User profiles and logged meals are stored in files to ensure data is available across sessions.
- The system reads and writes *JSON-formatted* data for easy retrieval.

**JSON Parsing & API Integration:**

- The system fetches nutritional data from the *Nutritionix* API using HTTP requests.
- API responses in JSON format are parsed to extract calorie and nutrient values.

**User Input Handling & Validation:**

- Users enter food names, portion sizes, and other details, which are validated to prevent errors.
- Input sanitization ensures correct API queries.

**Command-Line Interface (CLI) Design:**

- A text-based menu guides users through actions like logging food, viewing nutrition info, and checking calorie intake.
- The system ensures a structured and user-friendly CLI experience.

based on weight, height, age, and gender.

**Mifflin-St Jeor Equation for Calorie Calculation:**

- Estimates **Basal Metabolic Rate (BMR)* based on weight, height, age, and gender.
- Adjusts BMR based on activity level to calculate Total Daily Energy Expenditure (TDEE).
- Helps users determine calorie needs for maintenance, weight loss, or gain.

**Calorie Calculation & Meal Categorisation:**

- Calories are computed based on logged food items.
- Meals are categorized into Breakfast, Lunch, Snacks, and Dinner for better organization.

**Modular Programming & Code Organization**

- Separates functionalities into functions within *utils.py* for better maintainability.

**Error Handling & Exception Management**

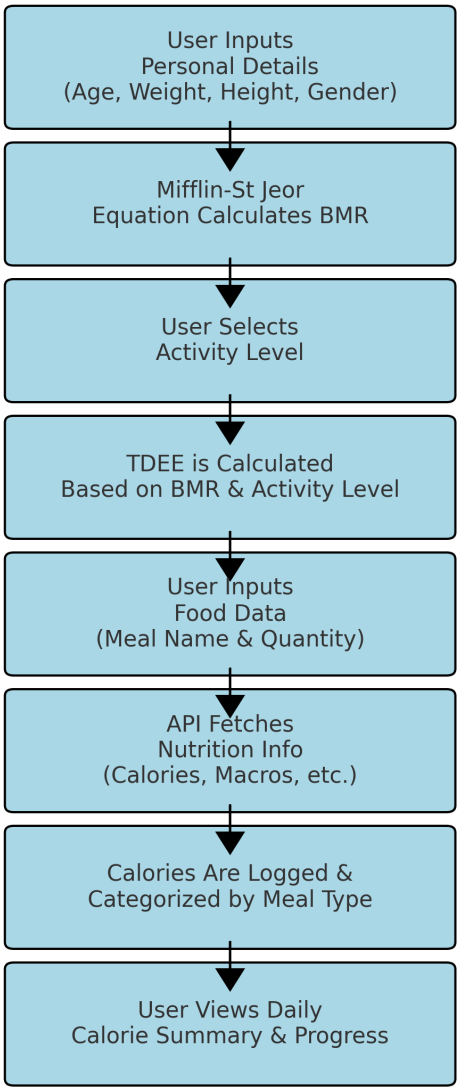
- Prevents crashes due to incorrect input, API failures, and file errors using try-except blocks.

**Basal Metabolic Rate (BMR) – The number of calories the body requires to maintain basic functions like breathing, circulation, and cell production while at complete rest. It represents the minimum energy expenditure needed to sustain life(more explanation here)*

# SYSTEM DESIGN & REQUIREMENTS

# 2.1 SYSTEM FLOW

The system follows a structured process:




**User Input Collection** – Age, weight, height, gender, and activity level are collected.

**BMR & TDEE Calculation** – Using the Mifflin-St Jeor Equation, the system determines Basal Metabolic Rate (BMR) and adjusts it based on activity level to get Total Daily Energy Expenditure (TDEE).

**Meal Logging –** User logs food, and nutritional data is retrieved from a database or API.

**Caloric Analysis** – The system calculates the total intake and updates remaining calorie balance.

**Summary Report –** Displays daily intake, remaining calories, and suggestions.

# 2.2 PREREQUISITES FOR RUNNING CODE

**1. Software Requirements**

- Python 3.x (Recommended: Python 3.8 or later)
- A code editor or IDE (e.g., VS Code, PyCharm, Jupyter Notebook)

**2. Required Python Libraries**

Install the following dependencies using pip:

```
pip install requests json5
```

**3. API Key**

- Obtain an API key by registering on Nutritionix API Website
- Store the API key securely in an environment variable or a configuration file.

**4. Internet Connection**

- Required for API requests to fetch food nutrition data.

# IMPLEMENTATION & DEMONSTATION

# 3.1 IMPLEMETATION DETAILS

The Diet Tracking System is developed using Python and follows a modular programming approach to ensure clarity and maintainability. The system integrates external APIs for food nutrition data and employs scientific formulas for Basal Metabolic Rate (BMR) and Total Daily Energy Expenditure (TDEE) calculations.

**User Input Handling:**

- The system collects user details such as age, weight, height, gender, and activity level to compute BMR and TDEE.
- Users log their daily food intake by providing meal names and quantities.

**BMR & TDEE Calculation:**

- The Mifflin-St Jeor Equation is used to calculate BMR, which estimates the minimum calories needed for bodily functions at rest.
- TDEE is then computed by multiplying BMR with an activity factor, determining the total calories required per day.

**Nutritional Data Retrieval:**

- The system integrates an external API (Nutritionix API) to fetch calorie and macronutrient details based on user-inputted food.
- API responses are processed to extract relevant nutritional data.

**Calorie Logging & Tracking:**

- The program categorizes food entries into meals (breakfast, lunch, dinner, snacks) and logs calorie consumption.
- Data is stored in JSON files for persistence.

**User Summary & Output:**

- The system displays a daily calorie summary, comparing calorie intake to TDEE to help users track their caloric surplus or deficit.
- Provides real-time feedback on diet progress.

## 3.1.1 EQUATION AND CALCULATIONS

The system relies on scientifically validated formulas to estimate energy needs and track nutrition effectively.

1. **Mifflin-St Jeor Equation (BMR Calculation):**

This equation calculates Basal Metabolic Rate (BMR), which is the number of calories a person needs to maintain basic bodily functions at rest.

For Men:

$$BMR = (10 \times \text{weight in kg}) + (6.25 \times \text{height in cm}) - (5 \times \text{age}) + 5$$

For Women:

$$BMR = (10 \times \text{weight in kg}) + (6.25 \times \text{height in cm}) - (5 \times \text{age}) - 161$$

**Example Calculation:** *(For a 70kg, 175cm, 25-year-old male)*

$$BMR = (10 \times 70) + (6.25 \times 175) - (5 \times 25) + 5 = 1{,}750 \text{ kcal/day}$$

2.**Total Daily Energy Expenditure (TDEE):**

Once BMR is determined, it is adjusted based on the user's activity level

$$TDEE = BMR \times Activity\ Factor$$

| Activity Level | Multiplier |
|---|---|
| Sedentary (little to no exercise) | 1.200 |
| Lightly active (1-3 days/week) | 1.372 |
| Moderately active (3-5 days/week) | 1.550 |
| Very active (6-7 days/week) | 1.725 |
| Super active (athlete, intense training) | 1.900 |

**Example:**

If a user's BMR is 1,750 kcal and they are moderately active, then:

$$TDEE=1,750\times1.55=2,712\ kcal/day$$

3.**Caloric Balance Formula:**

To determine how many calories remain for the day:

$$Remaining\ Calories=TDEE-Calories\ Consumed$$

# 3.2 CODE OVERVIEW

This project is designed with a modular structure, ensuring code reusability, maintainability, and scalability. The implementation is divided into two main files:

*utils.py* – Contains all helper functions for calculations and data handling.
*main.py* – Serves as the entry point, orchestrating the entire process.

*By keeping core logic separate from execution, the system remains flexible and easy to update or expand in the future.*

# 3.2.1 IMPLEMENTATIONOF UTILS.PY(UTILITIES)

```
1   import os
2   import json
3   from dotenv import load_dotenv
4   import requests
5
6   load_dotenv()
7
8   APP_ID = os.getenv("APP_ID")
9   API_KEY = os.getenv("API_KEY")
10  BASE_URL = "https://trackapi.nutritionix.com/v2/natural/nutrients"
11  USER_PROFILE_FILE = "user_profiles.json"
12
13  > def save_user_profile(userName, profile_data):…
24
25  > def load_user_profile(userName):…
32
33  > def get_user_log_file(userName):…
35
36  > def get_info(userName):…
101
102 > def get_nutrition(food_item):…
116
117 > def view_logs(userName):…
146
147 > def log_food(userName, meal_type, food_name, portion, calories):…
164
165 > def clear_screen():…
```

```python
import os
import json
from dotenv import load_dotenv
import requests

load_dotenv()

APP_ID = os.getenv("APP_ID")
API_KEY = os.getenv("API_KEY")
BASE_URL = "https://trackapi.nutritionix.com/v2/natural/nutrients"
USER_PROFILE_FILE = "user_profiles.json"

def save_user_profile(userName, profile_data):
    try:
        with open(USER_PROFILE_FILE, "r") as file:
            profiles = json.load(file)
    except (FileNotFoundError, json.JSONDecodeError):
        profiles = {}

    profiles[userName] = profile_data

    with open(USER_PROFILE_FILE, "w") as file:
        json.dump(profiles, file, indent=4)

def load_user_profile(userName):
    try:
        with open(USER_PROFILE_FILE, "r") as file:
            profiles = json.load(file)
        return profiles.get(userName)
    except(FileNotFoundError, json.JSONDecodeError):
        return None

def get_user_log_file(userName):
    return f"{userName}_diet_log.json"

def get_info(userName):
    existing_profile = load_user_profile(userName)
    if existing_profile:
        print("\n✅ Found existing profile. Using saved data")
        return existing_profile['calInt']

    while True:
        try:
            profile_data = {
                "weight": weight,
                "target": target,
                "time": time,
                "height": height,
                "age": age,
                "gender": gender,
                "actLvl": actLvl,
                "calIn": calIn
            }
            save_user_profile(userName, profile_data)

            return calIn

        except ValueError as e:
            print(f"❌ Error: {e}. Please Re-enter your inputs.")

def get_nutrition(food_item):
    headers = {…
    data = {"query": food_item}
    response = requests.post(BASE_URL, headers=headers, json=data)

    if response.status_code == 200:
        return response.json()
    else:
        print("Error fetching data:", response.text)
        return None

def view_logs(userName):
    log_file = get_user_log_file(userName)
    try:
        with open(log_file, "r") as file:
            logs = json.load(file)
    except (FileNotFoundError, json.JSONDecodeError):
        logs = {"Breakfast": [], "Lunch": [], "Snacks": [], "Dinner": []}

    print("\nFood Log:")

    total_calories_per_meal = {"Breakfast": 0, "Lunch": 0, "Snacks": 0, "Dinner": 0}
    grand_total_calories = 0

    for meal, items in logs.items():
```

```
    total_calories_per_meal = {"Breakfast": 0, "Lunch": 0, "Snacks": 0, "Dinner": 0}
    grand_total_calories = 0

    for meal, items in logs.items():
        print(f"\n{meal}:")
        if items:
            for entry in items:
                print(f"  - {entry['food']} ({entry['portion']}) - {entry['calories']} cal")
                total_calories_per_meal[meal] += entry['calories']
        else:
            print("No entries.")

    print("\n--- Total Calories per Meal ---")
    for meal, total in total_calories_per_meal.items():
        print(f"{meal}: {total} cal")
        grand_total_calories += total


    print("\n🔥 Grand Total Calories for the Day:", grand_total_calories, "cal")

def log_food(userName, meal_type, food_name, portion, calories):
    log_file = get_user_log_file(userName)
    try:
        with open(log_file, "r") as file:
            logs = json.load(file)
    except (FileNotFoundError, json.JSONDecodeError):
        logs = {"Breakfast": [], "Lunch": [], "Snacks": [], "Dinner": []}

    default_logs = {"Breakfast": [], "Lunch": [], "Snacks": [], "Dinner": []}
    logs = {**default_logs, **logs}

    logs[meal_type].append({"food": food_name, "portion": portion, "calories": calories})

    with open(log_file, "w") as file:
        json.dump(logs, file, indent=4)

    print(f"Logged: {food_name} ({portion}) - {calories} kcal for {meal_type}")

def clear_screen():
    os.system('cls' if os.name == 'nt' else 'clear')
```

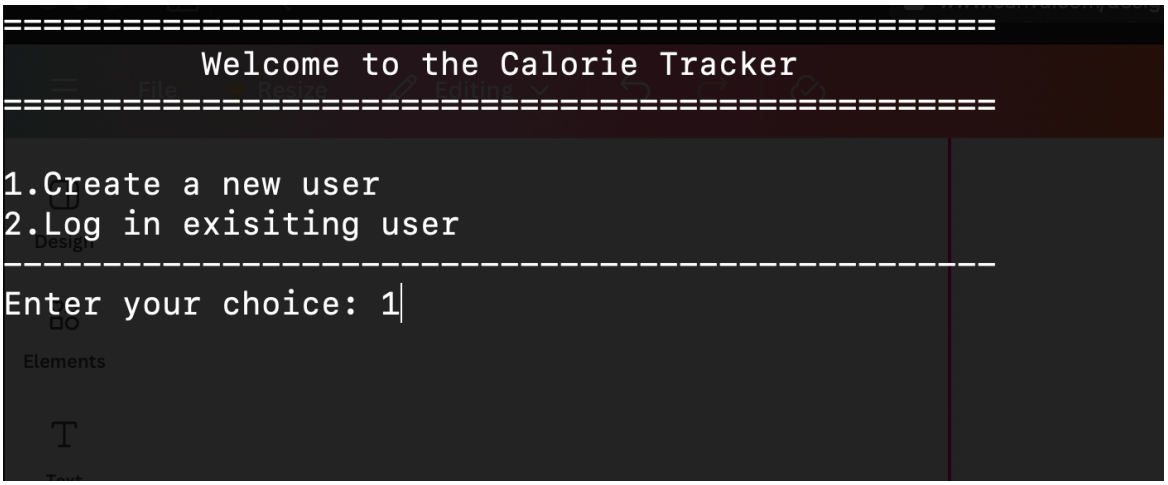
# 3.2.2 IMPLEMENTATION OF MAIN.PY

```python
import utils as ut
import os

if not os.path.exists(".env"):
    raise FileNotFoundError("❌ Missing .env file. Please create one with APP_ID and API_KEY.")

def main():
    while True:
        ut.clear_screen()
        print("=" * 50)
        print("          Welcome to the Calorie Tracker")
        print("=" * 50)

        print("\n1.Create a new user\n2.Log in exisiting user")
        print("-" * 50)
        while True:
            try:
                new_Choice = int(input("Enter your choice: "))
                if new_Choice not in [1, 2]:
                    raise ValueError("Invalid choice. Please enter 1 or 2.")
                break
            except ValueError as e:
                print(f"❌ Error: {e}")
        userName = input("\nEnter your username: ").strip()

        if new_Choice == 1:
            ut.clear_screen()
            if ut.load_user_profile(userName):
                confirm = input("⚠️ A profile with this username already exists. Overwrite? (yes/no): ").strip().lower()
                if confirm != "yes":
                    print("❌ Profile creation canceled.")
                continue
            calories = ut.get_info(userName)
            print(f"\n✅Your daily calorie intake should be: {calories} calories\n")
        elif new_Choice == 2:
            ut.clear_screen()
            profile = ut.load_user_profile(userName)
            if profile:
                calories = profile['calIn']
                print(f"\n✅Welcome back, {userName}! Hope your are doing well!\n"
                      f"As per your Previous data, your daily calorie intake should be {calories} calories")
                input("\nPress Enter to move to the main menu...")
            else:
                print("❌ No profile found for this username. Please create a new user.")
                input("\nPress Enter to return to the main menu...")
                continue

        while True:
            ut.clear_screen()
            print("\nMain Menu")
            print("-" * 50)
            print("1. View Food Nutrition\n2. Log Food\n3. View Log\n4. Exit")
            print("-" * 50)
            choice = int(input("Enter your choice: "))

            if choice == 1:
                ut.clear_screen()
                food = input("\nEnter food name: ").strip()
                portion = input("Enter portion size (e.g., 100g, 1 unit): ").strip()
                query = f"{portion} of {food}"
                data = ut.get_nutrition(query)
                if data and "foods" in data:
                    item = data["foods"][0]
                    print(
                        f"\n🍽 Nutrition Values of {item['food_name'].upper()} as per inputted portion of {portion.upper()}:\n"
                        f" - Calories: {item['nf_calories']} calories\n"
                        f" - Carbs    : {item['nf_total_carbohydrate']} grams\n"
                        f" - Proteins: {item['nf_protein']} grams\n"
                        f" - Fibers  : {item['nf_dietary_fiber']} grams\n"
                        f" - Fats    : {item['nf_total_fat']} grams"
                    )
                    input("\nPress Enter to return to the main menu...")

            elif choice == 2:
                ut.clear_screen()
                while True:
                    print("\nLog Food Menu")
                    print("-" * 50)
                    print("1. Breakfast\n2. Lunch\n3. Snacks\n4. Dinner\n5. Back to Main Menu")
                    print("-" * 50)
                    meal_choice = int(input("\nEnter meal type: "))
                    meal_map = {1: "Breakfast", 2: "Lunch", 3: "Snacks", 4: "Dinner"}
                    if meal_choice not in meal_map and meal_choice != 5:
                        print("❌ Invalid choice. Please try again.")
```

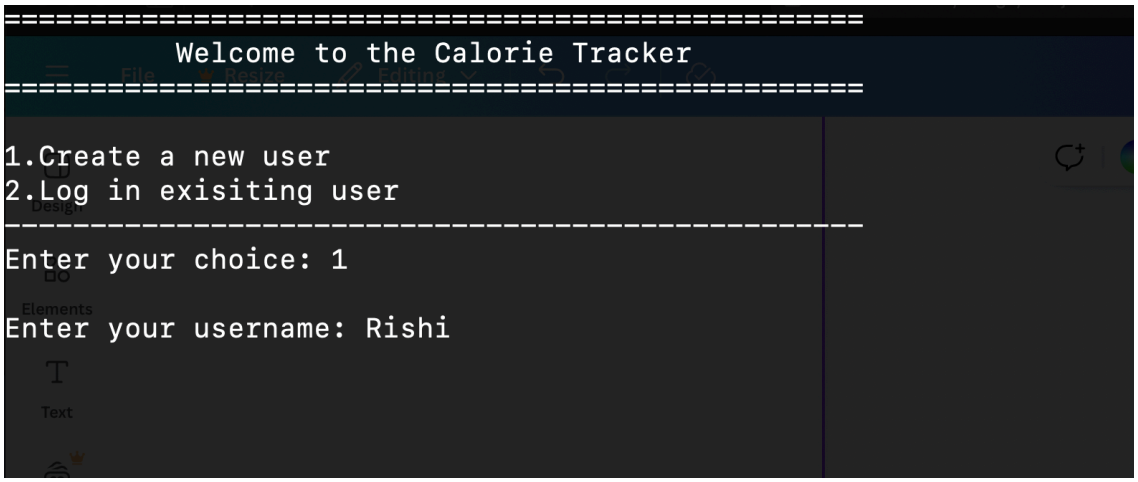
```
                    input("\nPress Enter to return to the log food menu...")
                    continue

                if meal_choice in meal_map:
                    meal_type = meal_map[meal_choice]
                    food = input("Enter food name: ").strip()
                    portion = input("Enter portion size (e.g., 100g, 1 unit): ").strip()
                    query = f"{portion} of {food}"
                    data = ut.get_nutrition(query)
                    if data and "foods" in data:
                        item = data["foods"][0]
                        ut.log_food(userName, meal_type, item['food_name'], portion, item['nf_calories'])
                        print(f"✅ Logged: {item['food_name']} ({portion}) - {item['nf_calories']} cal")
                    else:
                        print("⚠️ Error: Unable to fetch nutrition data.")
                        input("\nPress Enter to return to the main menu...")
                elif meal_choice == 5:
                    break
            elif choice == 3:
                ut.clear_screen()
                print("\n📜 Viewing Food Log...")
                ut.view_logs(userName)
                print(f"🔥 Your target Calories : {calories} cal")
                input("\nPress Enter to return to the main menu...")
            elif choice == 4:
                ut.clear_screen()
                print("\nThank you for using the Calorie Tracker. Goodbye!")
                print("=" * 50)
                exit()
            else:
                print("❌ Invalid choice. Please try again.")
                input("\nPress Enter to return to the main menu...")

if __name__ == "__main__":
    main()
```

# 3.3 CODE DEMONSTRARTION

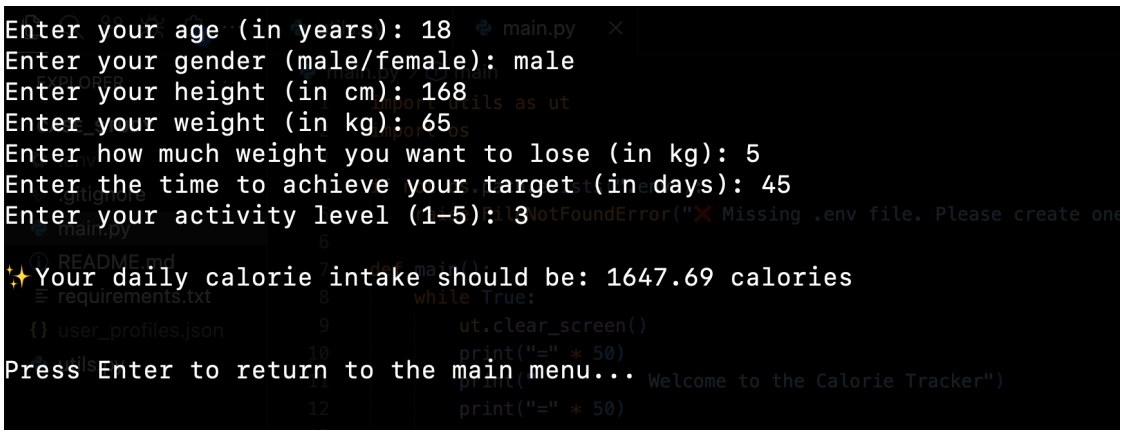
after running *python main.py* in your CLI, you will be greeted with this interface




here in example, *option 1 has been chosen* to create a new user.



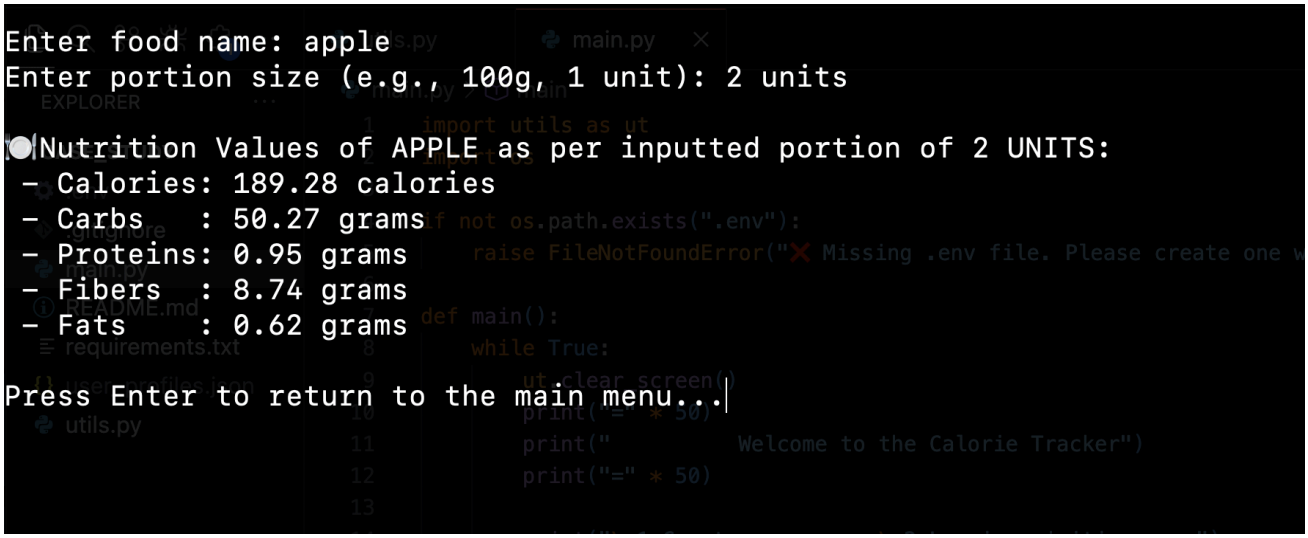

*If a user has previously logged their details, the system automatically retrieves and uses their saved profile data, eliminating the need to re-enter information.*




*After Hitting enter, we are asked to fill out the Bio-data for Calorie deficit calculations*


*After Entering Relavent info, user is greeted with a Menu screen which provides core fucntionality for user*

1. Viewing Nutritional Values




2. Logging Food

3. Viewing Food Logs


# Future Enhancements

To improve the functionality and user experience, the following features are planned for future updates:

**Water Logs:** Users will be able to track their daily water intake, ensuring they stay hydrated.
**Edit Bio Data:** Users will have the option to update their personal details, such as weight, height, and activity level, to keep their profile information accurate.

# HELPFUL SOURCES

*Mifflin-St Jeor Equation : https://www.jandonline.org/article/S0002-8223(05)00149-5/abstract*

*Source Repository: https://github.com/Pengw0in/Diet-Tracking-System--CLI-Based-*

# THANK YOU